Deep Learning Project 2

Pranav A (20478966)
Big Data Institute
Hong Kong University of Science and Technology

November 24, 2017

Abstract

In this project, I have used convolutional neural networks and ResNet as the classification of images. At the end, ResNet performed better than CNNs. CNNs performed around 66% on validation set but ResNet scored around 93% on the dataset. Poor performance of ConvNet is due to extreme overfitting despite of application of dropout and batch normalization to this.

1 Introduction

This project was about the image classification. We were asked to built an image classifier based on the set of images of flowers and classify them in 5 classes.

Convolutional Neural Networks are a prevalent architecture for the image classification. ConvNets often involve convolutions over a set of filters, followed by pooling and usually it ends with a fully connected layer.

However, it is rare to get good accuracy from training with ConvNets from scratch. Thus, it is preferable to download a pretrained model, modify the last layers and finetune the weights on the given dataset.

Here, I used ResNets (Residual Networks) as my choice for the pretrained model. ResNet makes use of skip connections in their networks to learn the residuals or the "deficiencies" of the training. Following sections will show how I did pre-processing of data and used these two models in this project.

2 Experimental Setup

The given data was divided into training, validation and test sets. I chose to follow ImageNet conventions to preprocess the image. The data was resized to 224 x 224 pixels. For the 3 colour channels (RGB), I normalized that according to the $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$. Then I did certain data augmentation techniques like horizontal flipping and random cropping and zooming. I have used Pytorch [**?**] for developing the architectures.

2.1 CNN architecture details

For the first layer of CNN, I used 16 filters with the size of 3 x 3, padding of 1 with 0-length strides. Followed by that I used batch normalization over the stacked layer, ReLU activation and max pooling with a stride of 2.

For the second layer of CNN, I used 32 filters with the size of 5 x 5, padding of 2 with 0 strides. Followed by that I used batch normalization over the stacked layer, ReLU activation and max pooling with a stride of 2.

Followed by that, I used 3 fully connected layers (20000 x 500, 500 x 50, 50 x 5) with a dropout of 75% and ReLU activation. Finally, I took the log of the softmax of the resulting array.

Adam has been used as the optimizer with the learning rate of 0.001 and Cross Entropy as the loss function. The batch size is 64 with the number of epochs around 400.

2.2 ResNet transferred details

Variant of ResNet with 18 layers was used for this project. I added a layer on the top of the pretrained model and changed it to 5 classes. Stochastic Gradient Descent has been used as the optimizer with the learning rate of 0.001 and Cross Entropy as the loss function after taking the softmax of the output. The batch size is 64 with the number of epochs around 40.

3 Final Remarks

The results are shown in Table 1. Clearly, ResNet performs better than CNN.

	Training Accuracy	Validation Accuracy
CNN	**98%**	66%
ResNet-18	93.4%	**92.4%**

Table 1: Accuracy results of the algorithms

CNN was heavily overfitting despite applying batch normalization and dropout to it. Transfer Learning based architectures often act as the regularizer. Hence, no overfitting resulted in training of the ResNet.